How Reliable is My Wearable: A Fuzz Testing-based Study



Edgardo Barsallo Yi, Amiya Maji, Saurabh Bagchi

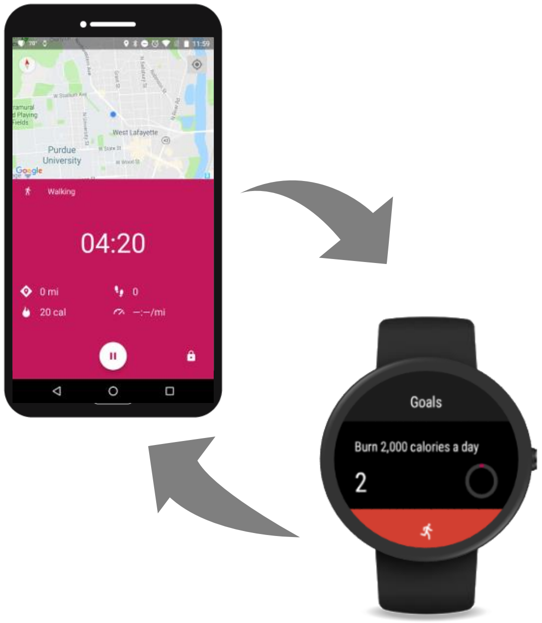
Purdue University

Outline

- Motivation and Background
- Android and Android Wear Overview
- Approach to Evaluate Reliability: Qui-Gon Jinn (QGJ)
- Evaluation
- Conclusion and Lessons Learned

Motivation

- Reliability of Android is well explored but wearables come with a new set of challenges
- Wearable devices are **sensor rich**
- Devices have **limited resources**
  - Display area, computing power, volatile and non-volatile memory, battery size
- **More background work** (services) than foreground work (activities)
- **Communication pattern** where many apps are controlled by a mobile counterpart
- A large use-case is monitoring, accumulation and dissemination of **health and fitness** data

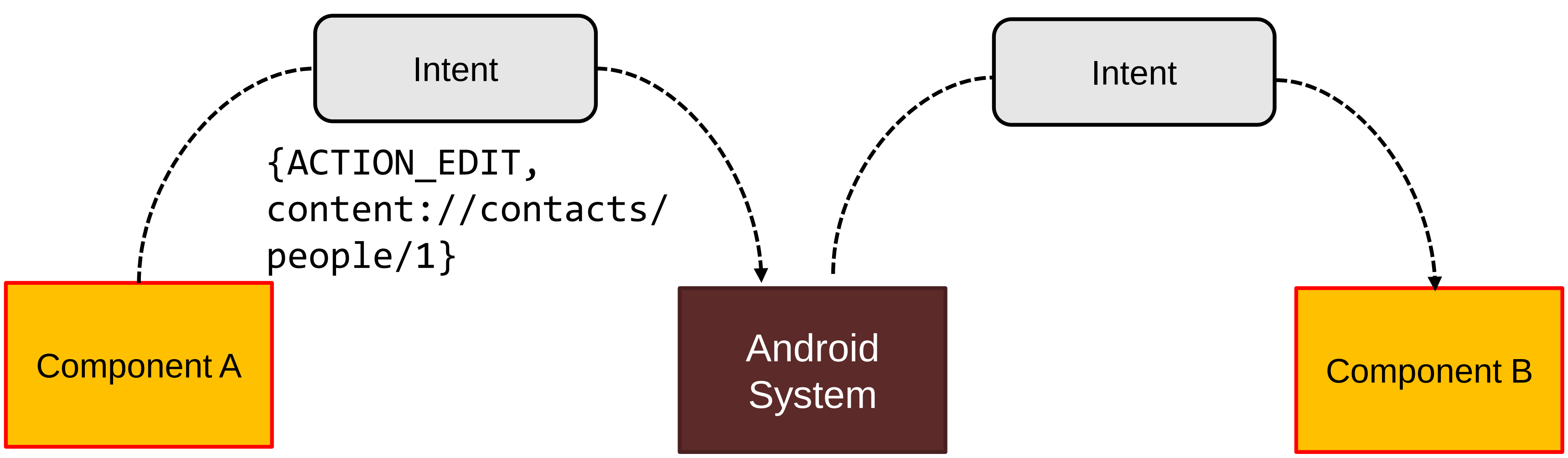


Android Wear (AW)

- The most popular wearable OS (released in 2014): We use 2017 release
- User Interface (UI) is designed **to require minimal human interaction** (micro transactions)
- Applications are more **services driven**, in contrast to Android applications, which usually have rich GUI
- AW makes heavy use of sensors
  - Common use case of fitness and health monitoring

Android IPC: Background

- Android programming model is based on passing **Intent** messages among the components.
- An **Intent** describes an operation to be performed and data to perform it
  - The basic information includes: Action, Category, Data, Component, and Extras
  - Types: Explicit and Implicit



Outline

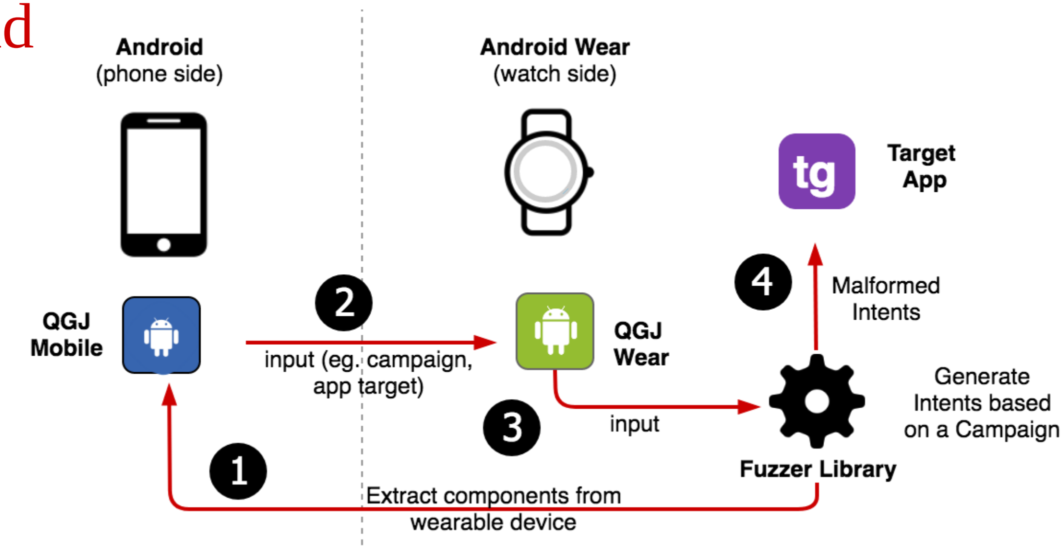
- Motivation and Background
- Approach to Evaluate Reliability: *Qui-Gon Jinn* (QGJ)
- Evaluation
- Conclusion and Lessons Learned

Approach to Evaluate Reliability

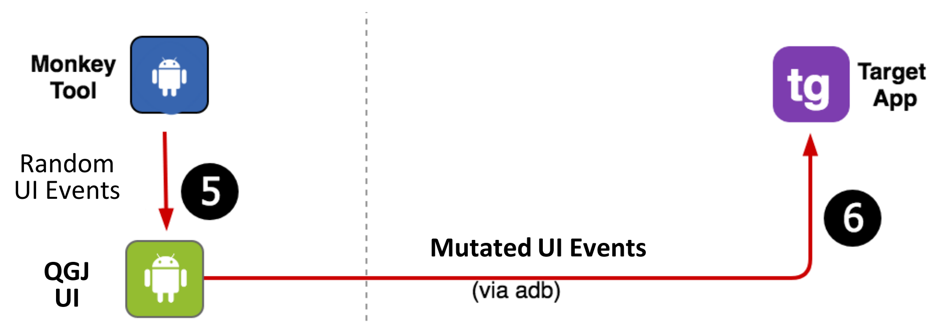
- Evaluate robustness of Android Wear apps by injecting fuzzed Intents
- Discover vulnerabilities through random and mutated Intents
- Propose recommendations for improving the robustness of Android Wear apps

Qui-Gon Jinn (QGJ)

- QGJ is a user-level app and does not require any system-level permission
- QGJ-Master
  - Generates intents based on Fuzz Injection Campaigns (FIC)
  - Supports fuzz injection of Activities and Services components
- QGJ-UI
  - Mutates semi-valid and random UI events
  - Injects UI events using Android Debug Bridge (adb)



QGJ-Master



QGJ-UI

PURDUE UNIVERSITY

Fuzz Injection Campaigns

- Semi-valid Action and Data

{act=ACTION_EDIT, dat=file://sdcard/file.txt, cmp=some.component.name} ✓	{act=ACTION_DIAL, data=file://sdcard/file.txt, cmp=some.component.name} FUZZED

- Blank Action or Data

{act=ACTION_VIEW, dat=https://youtu.be/j5dMnAP242Z, cmp=some.component.name} ✓	{dat=https://youtu.be/j5dMnAP242Z, cmp=some.component.name} FUZZED

- Random Action or Data

{act=ACTION_EDIT, dat=content://contacts/people/1, cmp=com.android.contacts} ✓	{act=ACTION_EDIT, dat=q1w2e3Q!W@E#, cmp=com.android.contacts} FUZZED

- Random Extras

{act=ACTION_INSERT, dat=content://contacts/people/1, cmp=com.android.contacts (has extras)} ✓	{act=ACTION_INSERT, data=content://contacts/people/1, cmp=com.android.contacts (has random extras)} FUZZED

Outline

- Motivation and Background
- Approach to Evaluate Reliability: *Qui-Gon Jinn* (QGJ)
- Evaluation
- Conclusion and Lessons Learned

Target Applications

- **Categories**: Health/Fitness and Not Health/Fitness
  - Based on the fact that health/fitness tracking apps are prominent in AW ecosystem (use of hardware and software sensors)
- **Classification**: Built-in and Third-party apps
  - Built-in apps are pre-installed on the wearable device, while third-party apps are installed by the user
- **Maturity Level:** Third-party apps with at least 1M downloads from the Google Play Store

- **Comparison between Android and AW Ecosystem**:
  - We conducted similar experiments on Android using QGJ-Main
  - We focused on Android built-in apps which are often used by third-party application for implementing common functionalities (**com.android**)

Experimental Configuration

- QGJ-Master
  - Android 7.1.1 (released Dec 2016)
  - Android Wear 2.0 (released Feb 2017)
- QGJ-UI
  - Android Wear 2.0 (Emulator)
- Applications

OS		Classification	#	# Activities	# Services
AW	Health/Fitness	Built-in	2	81	34
AW	Health/Fitness	Third Party	11	80	59
AW	No Health/Fitness	Built-in	9	168	188
AW	No Health/Fitness	Third Party	24	185	117
AW	**Total**		**46**	**514**	**398**
A	**com.android**	**Built-in**	**63**	**595**	**218**

Error Manifestations

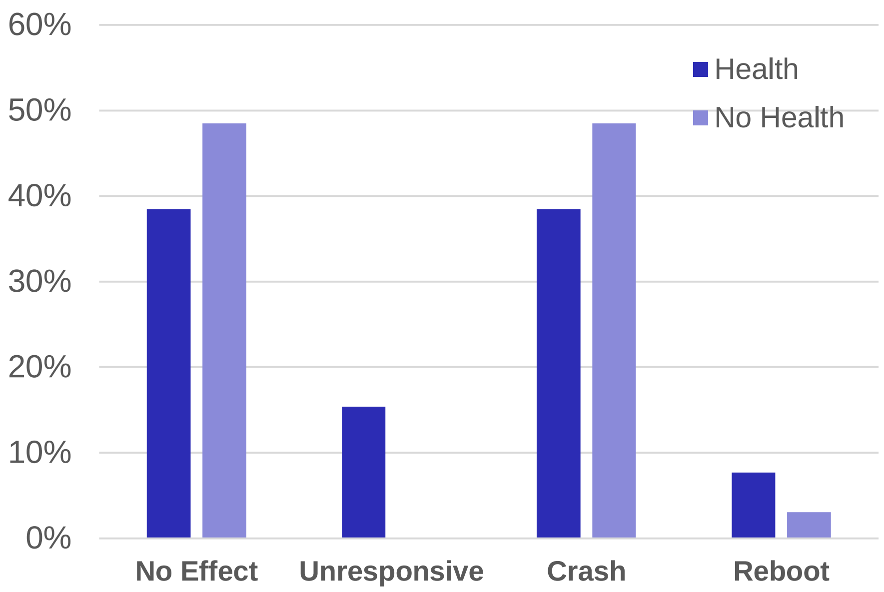
- System Reboot
  - The OS reaches an unrecoverable state and the device reboots
- Crash
  - Application crashes due inability to handle malformed intents
- Hang or unresponsive
  - The application experiences temporary unresponsiveness or freezes permanently
- No effect
  - No effect or failure manifestation due to the malformed injection

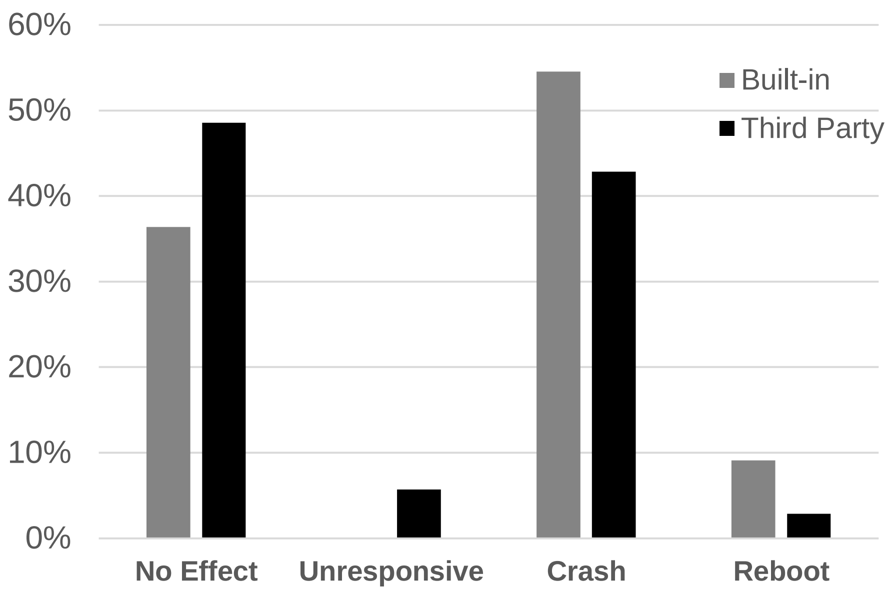
Distribution of Behaviors Among Fuzz Intent Campaigns

	Reboot		Crash		Hang		No Effect	
	Health	No Health	Health	No Health	Health	No Health	Health	No Health
Semi-valid	8%	0%	23%	30%	8%	0%	62%	70%
Blank Action or Data	0%	0%	31%	24%	0%	0%	69%	76%
Random Action or Data	0%	0%	31%	33%	8%	0%	62%	67%
Random Extras	0%	3%	15%	30%	8%	0%	77%	67%

- System Reboots occurred on both categories
- Injection has no effect at roughly the same rate (~70%) on both categories

Reliability per Category



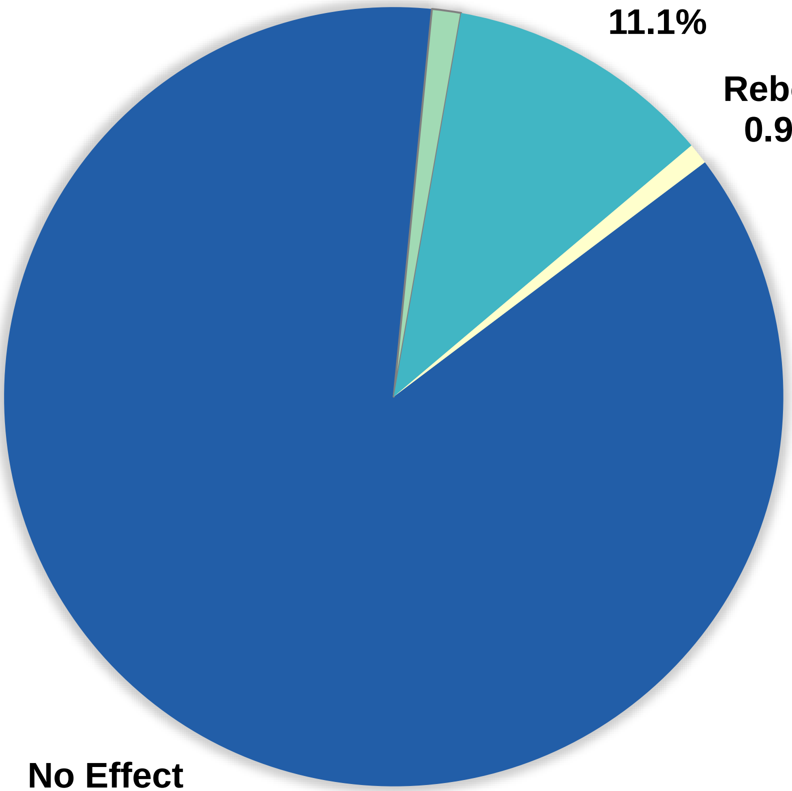


No significant difference between **Health/Fitness** apps and other apps

Built-in applications showed more failures compared to **Third Party** apps

PURDUE UNIVERSITY

How Reliable is the Wearable Software Stack?



- Distribution of error manifestation among the application components
  - 13% of components reported failures
  - Crashes are more common than unresponsive manifestation (9X)
  - System reboots affected less than 1% of components

PURDUE UNIVERSITY

Distribution of Crashes

75%

Exception	#Crashes	%
NullPointerException	54	30.9%
ClassNotFoundException	46	26.3%
IllegalArgumentException	31	17.7%
IllegalStateException	10	5.7%
RuntimeException	9	5.1%
ActivityNotFoundException	7	4.0%
UnsupportedOperationException	6	3.4%
Others	12	6.9%

Android

73%

Exception	#Crashes	%
NullPointerException	42	53.2%
IllegalStateException	10	12.7%
IllegalArgumentException	9	11.4%
ActivityNotFoundException	4	5.1%
Exception	4	5.1%
WindowManager$BadTokenException	3	3.8%
ClassNotFoundException	3	3.8%
Others	4	5.1%

Android Wear

Resilience against UI injection

Experiment	#Injected Events	Exceptions Raised	Crashes
Semi-valid	41405	1496 (3.6%)	22 (0.05%)
Random	41405	615 (1.5%)	0 (0%)

- QGJ-Master focus on the communication between components (either starting an Activity or a Service)
  - After, an activity or service has been started, some user interaction (UI events) take places.
  - QGJ-UI emulates this interaction to test the robustness of apps
- No system crash during the UI injection
- Fewer number of exceptions and crashes than QGJ-Master
  - QGJ-UI only injects events to launcher activities
  - adb tools have a robust input validation

System Crashes from User-level Application

- No extra permissions at install time
- The manifestation depends on the transient state of the device
  - The reboots were not triggered by single intent, but due to error propagation across components and software aging through repeated fuzzing campaigns.
- 2 apps crashed Android Runtime
  - A health app (third party) raised a **SIGABRT** signal during the experiment, after experiencing some unresponsiveness.
  - A built-in app raised a **SIGSEVG** signal. The app crashed multiple times during the injection before triggering the reboot.

Outline

- Motivation and Background
- Approach to Evaluate Reliability: *Qui-Gon Jinn* (QGJ)
- Evaluation
- Conclusion and Lessons Learned

Conclusions and Insights

- **Distribution of exception** types differ between Android and Android Wear
  - Input validation (e.g. NullPointerException) is still the major cause of crashes
  - High incidence of crashes on AW are tied to the state of the application/device (e.g. IllegalStateException)
- **Software Aging**: Further research on software aging can help identify and mitigate transient system reboots that are state dependent
- **Input Validation**: Although Android's input validation has improved compared to earlier work [Maji, DSN'12] it is still a major cause for crashes.
  - Need more awareness and tool support